Leaves classification

Chiara Saini

Abstract

The following analysis aims to explore Convolutional Neural Networks for image recognition. The analysed images belong to twenty two classes of sane and ill leaves. Firstly, the data-set, the image preprocessing, the train data and the class weights will be explained. After, Convolutional neural network models used will be described. The first model was created as a benchmark. The second model was built using a complex hyperparameter, and architecture tuning technique.The third model has different architecture, and a different input shape than the others.

To realize this analysis, I have consulted TensorFlow, Keras and sklearn documentation.

1 The dataset

The dataset is composed by RGB images of leaves belonging to different plants species.

RGB images are composed of three independent color channels: red, green, and blue. Together these reproduce a set of arrays of colors that will give a colored image as output. An RGB image will be three of the two-dimensional matrix concatenated to each other, one for each channel of the color. In Figure 1 it is possible to see an RGB image representation[1].

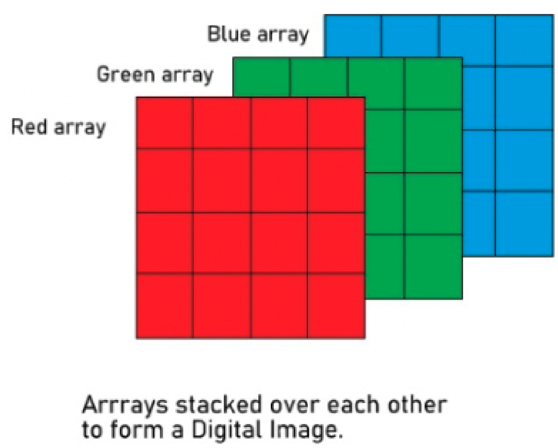

Figure 1: Example of RGB image representation.

To retrieve the data from the directory, the method `tf.keras.utils._image_dataset_from_directory` was used, the batch size was set to 32, the image width and hight was set to 180, the label was set to "inferred" and the shuffle to "True".

The dataset is already divided in train, validation and test. The train dataset is composed of 4274 files belonging to 22 classes, the validation dataset is composed of 110 files belonging to 22 classes, and the test dataset is composed of 110 files belonging to 22 classes. All images are in ".jpg" format. Half of the classes are composed by healthy leaves and the other half are unhealthy. There were no corrupted images, so every file was used in the analysis. The classes inside each dataset are:

[1]Reference: "Principal Component Analysis For Image Data in Python", Ask Python, https://www.askpython.com/python/examples/principal-component-analysis-for-image-data

1. Alstonia Scholaris diseased;
2. Alstonia Scholaris healthy;
3. Arjun diseased;
4. Arjun healthy;
5. Bael diseased;
6. Basil healthy;
7. Chinar diseased;
8. Chinar healthy;
9. Gauva diseased;
10. Gauva healthy;
11. Jamun diseased;
12. Jamun healthy;
13. Jatropha diseased;
14. Jatropha healthy;
15. Lemon diseased;
16. Lemon healthy;
17. Mango diseased;
18. Mango healthy;
19. Pomegranate diseased;
20. Pomegranate healthy;
21. Pongamia Pinnata diseased;
22. Pongamia Pinnata healthy;

It is possible to see a representation of each class in Figure 2

Figure 2: Dataset classes.

The train dataset is highly unbalanced. It is possible to see the count of each class in Figure 3. Note that each name of the classes has been encoded, therefore the graph's labels corresponds to encoded classes.

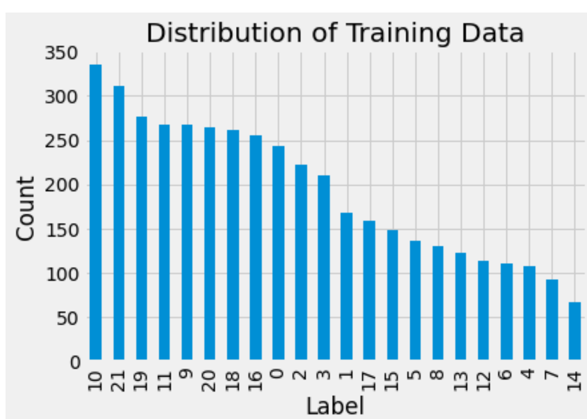

Figure 3: Distribution of classes in training set.

The classes in the validation and test set are balanced, as displayed in Figure4 and 5 respectively.

Given the imbalance of the trainig set, weights to each class were assigned in order to avoid bias. It is possible to see the weights corresponding to each encoded class in table 1.

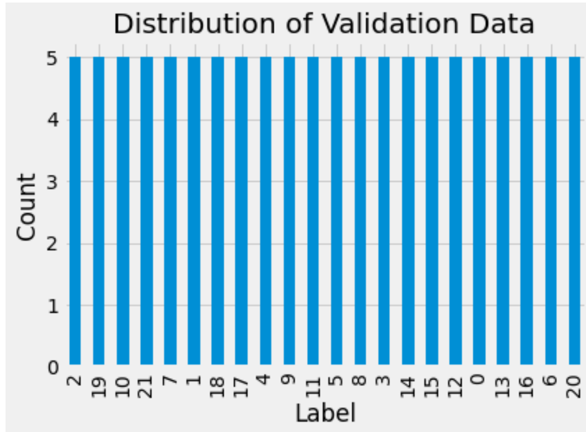

Figure 4: Distribution of classes in validation set.

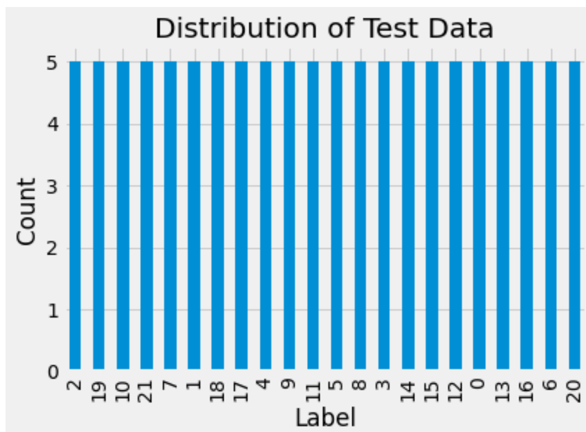

Figure 5: Distribution of classes in test set.

Table 1

Class	Weight	Class	Weight
0	0.7961997019374069	11	0.7248982360922659
1	1.1563852813852813	12	1.70414673046252
2	0.8751023751023751	13	1.5794530672579452
3	0.9251082251082251	14	2.8995929443690636
4	1.8156329651656755	15	1.303843807199512
5	1.418049104180491	16	0.7618538324420677
6	1.7661157024793388	17	1.2218410520297314
7	2.088954056695992	18	0.744339951236503
8	1.482997918112422	19	0.7013455858221201
9	0.7276132107592782	20	0.7331046312178388
10	0.5799185888738128	21	0.6226689976689976

2 Convolutional Neural Network Theory

Neural Networks (NNs) are a type of directed graph whose nodes correspond to neurons and edges correspond to links between them. Each neuron receives as input a weighted sum of the outputs of the neurons connected to its incoming edges. Convolutional Neural Networks (CNNs) are one of the most used NNs architectures. CNNs are used to recognize visual patterns and features directly from pixel images with a certain degree of variability. The identification of patterns and features is possible thanks to the ability of CNNs to keep important information about the data analyzed. This is done through its special architecture, which will be explained in the following sections. The complexity of CNN increases with each layer. The most superficial layers focus on simple features like colors and edges. As the image data progresses through deeper layers of the CNN, the algorithm starts to recognize more significant elements or shapes of the object until it finally identifies the intended object. The layers in a CNN can be:

- **Convolutional layers (Conv2D)**: Convolutional layers uses filters to perform a convolutional operation as it analyzes the input with respect to its dimensions. In doing so, the features belonging to different classes will be detected and extracted from the image using the Kernel. A convolution layer consists of several convolution channels (depth or filters). Convolution channels are represented by numbers (such as 16, 32, 64, 128, etc.) that are equal to the number of channels in the output of a convolutional layer. The Kernel is the size of the convolution filters. Convolution filters remove unnecessary data and pull features from the input image. The Kernel is a matrix (in this case, a 3x3 matrix) that moves over the input data by a stride (in this case, it is a pixel unit), performing the dot product with a sub-region of input data. In this way, the network will learn through the filters that activate when detecting a specific feature in the spatial position of the input. The output of the convolutional layer is a feature map. To add complexity, more than one convolutional layer can be applied. Padding "same" is used, which the addiction of pixels needed for the convolutional Kernel onto the edge of the image. In this way, it is possible to solve the border effect. Finally the activation function used is the "relu", in order to transform the input values of neurons.

- **Pooling layers (MaxPooling2D)**: Pooling layers are required to down-sample the feature maps. This is done by summarizing the features in patches of the feature map. In the matter of this analysis, max pooling is used.

- **Flatten layers (Flatten)**: A Flatten layer flattens a multi-dimensional input into a single dimension.

- **Dropout layers (Dropout)**: A Dropout layer is used to prevent overfitting by randomly setting input units to 0 with a determined frequency rate. In this case the dropout rate is set to 0.2.

- **Dense layers (Dense)**: A Dense layer is deeply connected with its preceding layer because it receives its output.

- **Activation function layer (Softmax)**: Activation function can either be passed through an Activation layer, or through an activation argument [1]. In the case of this analysis an Activation layer is used, and the activation function is Softmax.

3 Models

3.1 Model 1

Model 1 is the first model of the analysis.

3.1.1 Architecture

Model 1 is a sequential model. The number of layers is 10 and the parameters in the model are 2.878.902.

1. **Convolutional layer**: the size of the filter is 16, the Kernel is 7x7, the input shape is defined (180x180, rgb image), the padding is set to "same", and the activation function is "relu";

2. **Max Pooling layer**: standard measures are left for this layer;

3. **Convolutional layer**: the size of the filter is 32, the Kernel is 5x5, the input shape is defined (180x180, rgb image), the padding is set to "same", and the activation function is "relu";

4. **Max Pooling layer**: standard measures are left for this layer;

[1] Reference: Layer activation functions, Keras documentation, https://keras.io/api/layers/activations/, last consultation: 14 March 2023

5. **Dense layer**: the size of the filter is 128, and the activation function is "relu";
6. **Dropout layer**: the dropout rate is set to 0.2;
7. **Dense layer**: the size of the filter is 64, and the activation function is "relu";
8. **Flatten layer**;
9. **Final Dense layer**: the size of the filter is 22;
10. **Activation function layer**: Softmax.

In figure 6 it is possible to see the summary of *Model 1*.

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_26 (Conv2D)          (None, 180, 180, 16)      2368

 max_pooling2d_30 (MaxPoolin  (None, 90, 90, 16)       0
 g2D)

 conv2d_27 (Conv2D)          (None, 90, 90, 32)        12832

 max_pooling2d_31 (MaxPoolin  (None, 45, 45, 32)       0
 g2D)

 dense_19 (Dense)            (None, 45, 45, 128)       4224

 dropout_5 (Dropout)         (None, 45, 45, 128)       0

 dense_20 (Dense)            (None, 45, 45, 64)        8256

 flatten_9 (Flatten)         (None, 129600)            0

 dense_21 (Dense)            (None, 22)                2851222

 softmax_1 (Softmax)         (None, 22)                0

=================================================================
Total params: 2,878,902
Trainable params: 2,878,902
Non-trainable params: 0
```

Figure 6: Summary of *Model 1*.

3.1.2 Compiling and fitting

The neural network was compiled as follows:

- **Optimizer**: Adam;
- **Loss**: Sparse Categorical Cross-entropy;
- **Metrics**: Accuracy.

The model fit was performed on the training set, and the the validation set. The epochs evaluated were 10, the batch size was set to 32, the callbacks were Reduce learning rate On Plateau, and early stopping.

3.1.3 Loss curves and Accuracy curves

The plot represented in Figure 7 shows that the accuracy and the loss curves for both train and validation set. It is possible to see that the model is highly overfitting. From epoch 3 on, the validation accuracy stops increasing. The validation loss is more or less constant through the epochs.

3.1.4 Prediction skills

Model 1 prediction skills were tested with the test set, and the f1 score resulted in 0.35681263181263184. This result is not surprising, given what observed in the previous section.

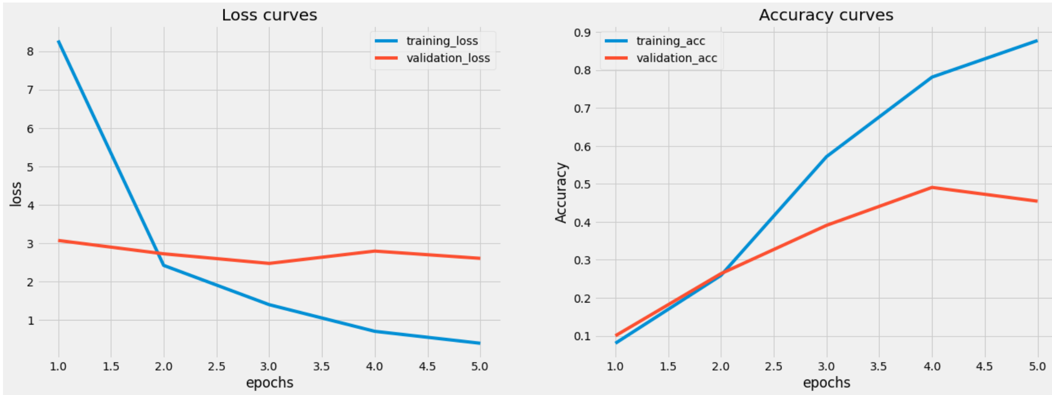

Figure 7: Loss and accuracy curves of *Model 1*.

3.2 Data augmentation

In order to improve model performances, the following layers were added for data augmentation:

- **RandomFlip**: two RandomFlip layers were used to perform horizontal and vertical flipping;
- **RandomRotation**: one RandomRotation layer was added to rotate the images by 0.5 units;
- **RandomZoom**: one RandomZoom layer was used to zoom the images by 0.5 units;
- **Rescaling**: the riscale parameter was 1/255.

3.3 Models with hyperparmeter and architecture tuning

The hyperparameter tuning was performed to choose the best amount of Convolutional and Dense layers, the best Convolutional filter and the best choice for the Dense layer. *Tensorboard* extension was used as a tool to manage the investigation of the best model.

The tested architectures were:

1. 3 convolutional layers, 32 nodes,0 dense layer, 1678879196 parameters;
2. 3 convolutional layers, 64 nodes, 0 dense layer, 1678879196 parameters;
3. 3 convolutional layers, 96 nodes, 0 dense layer, 1678879196 parameters;
4. 3 convolutional layers, 128 nodes, 0 dense layer, 1678879196 parameters;
5. 3 convolutional layers, 160 nodes, 0 dense layer, 1678879196 parameters;
6. 3 convolutional layers, 192 nodes, 0 dense layer, 1678879196 parameters;
7. 3 convolutional layers, 224 nodes, 0 dense layer, 1678879196 parameters;
8. 3 convolutional layers, 32 nodes,1 dense layer, 1678879196 parameters;
9. 3 convolutional layers, 64 nodes, 1 dense layer, 1678879196 parameters;
10. 3 convolutional layers, 96 nodes, 1 dense layer, 1678879196 parameters;
11. 3 convolutional layers, 128 nodes, 1 dense layer, 1678879196 parameters;
12. 3 convolutional layers, 160 nodes, 1 dense layer, 1678879196 parameters;
13. 3 convolutional layers, 192 nodes, 1 dense layer, 1678879196 parameters;
14. 3 convolutional layers, 224 nodes, 1 dense layer, 1678879196 parameters;
15. 3 convolutional layers, 32 nodes, 2 dense layer, 1678879196 parameters;

16. 3 convolutional layers, 64 nodes, 2 dense layer, 1678879196 parameters;
17. 3 convolutional layers, 96 nodes, 2 dense layer, 1678879196 parameters;
18. 3 convolutional layers, 160 nodes, 2 dense layer, 1678879196 parameters;
19. 3 convolutional layers, 128 nodes, 2 dense layer, 1678879196 parameters;
20. 3 convolutional layers, 192 nodes, 2 dense layer, 1678879196 parameters;
21. 3 convolutional layers, 224 nodes, 2 dense layer, 1678879196 parameters;
22. 3 convolutional layers, 32 nodes, 3 dense layer, 1678879196 parameters;
23. 3 convolutional layers, 64 nodes, 3 dense layer, 1678879196 parameters;
24. 3 convolutional layers, 96 nodes, 3 dense layer, 1678879196 parameters;
25. 3 convolutional layers, 128 nodes, 3 dense layer, 1678879196 parameters;
26. 3 convolutional layers, 160 nodes, 3 dense layer, 1678879196 parameters;
27. 3 convolutional layers, 192 nodes, 3 dense layer, 1678879196 parameters;
28. 3 convolutional layers, 224 nodes, 3 dense layer, 1678879196 parameters;

The activation functions for the Convolutional layers and the dense layers are the same as for Model 1, and in the Max Pooling layers the pooling size is set to 0.2.

Tensorboard was used to visualize the curves of loss and accuracy of training and validation. It is possible to consult all the experiments listed before at this page, otherwise the extended URL will be in the Appendix.

The best models form the tuning are listed below, together with the respective values of train, and validation accuracy, and train, and validation loss:

- 3 convolutional layers, 160 nodes, 1 dense layer, 1678879196 parameters;
  - Train Accuracy: 0.7473;
  - Train Loss: 0.02962;
  - Validation Accuracy: 0.6545;
  - Validation Loss: 1.441;
- 3 convolutional layers, 192 nodes, 1 dense layer, 1678879196 parameters;
  - Train Accuracy:0.7382;
  - Train Loss: 0.0307;
  - Validation Accuracy: 0.7091;
  - Validation Loss: 1.188;
- 3 convolutional layers, 224 nodes, 2 dense layer, 1678879196 parameters;
  - Train Accuracy: 0.7527;
  - Train Loss: 0.02775;
  - Validation Accuracy: 0.7091;
  - Validation Loss:1.099;
- 3 convolutional layers, 96 nodes, 1 dense layer, 1678879196 parameters;
  - Train Accuracy: 0.7361;
  - Train Loss: 0.03128;
  - Validation Accuracy: 0.6636;
  - Validation Loss: 1.49;

In Figure8 and 9 it is possible to the performances of the best performing models, listed above. Note that the legend of Figure8 and 9 is in Figure10 and 11.

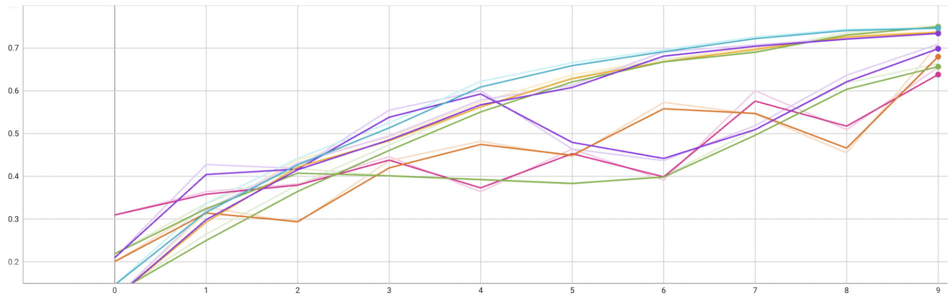

Figure 8: Accuracy curves of the best performing models, Train and Validation.

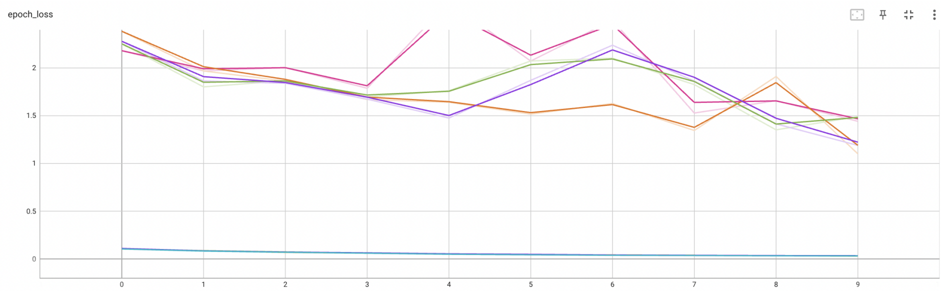

Figure 9: Loss curves of the best performing model, Train and Validation.

3.3.1 Compiling and fitting

The parameters for the compilation and fitting of the models are the same as for Model 1, except for callbacks that are set to "Tensorboard".

3.3.2 Loss curves and Accuracy curves

Given the models listed previously, the second one was chosen (3 convolutional layers, 192 nodes, 1 dense layer, 1678879196 parameters), and it was called **Model t1**. The total number of layers of Model t1 is 19. It is possible to see the summary of Model t1 in 12. During the model fitting phase, the model checkpoint callback, and the aforementioned class weights were added.

It is possible to better visualize the Loss and Accuracy curves of Model t1 in 13.

The plots in Figure 13 shows that the accuracy curves of train and validation tend be closer to each other at the end of the epochs with respect to the previous models. Overall the accuracy value does not represent a satisfying result, since from epoch 4 on there is a clear over-fitting. The validation loss curves seem to perform poorly.

3.3.3 Prediction skills

Model t1 prediction skills were also tested with the test set, and the f1 score resulted in 0.5225007315916407. In Figure 14 it is possible to see a comparison between the labels predicted by Model t1 and the true labels. It is safe to say that the algorithm does not predict well. These results were expected since the validation performance.

Run	Smoothed	Value	Step	Time	Relative
3-conv-160-nodes-1-dense-1678811835/train	0.7466	0.7473	9	3/14/23, 5:45 PM	7.252 min
3-conv-160-nodes-1-dense-1678811835/validation	0.6381	0.6545	9	3/14/23, 5:45 PM	7.252 min
3-conv-192-nodes-1-dense-1678812323/train	0.7367	0.7382	9	3/14/23, 5:53 PM	7.28 min
3-conv-192-nodes-1-dense-1678812323/validation	0.6985	0.7091	9	3/14/23, 5:53 PM	7.28 min
3-conv-224-nodes-2-dense-1678816225/train	0.7501	0.7527	9	3/14/23, 6:58 PM	7.279 min
3-conv-224-nodes-2-dense-1678816225/validation	0.6799	0.7091	9	3/14/23, 6:58 PM	7.279 min
3-conv-96-nodes-1-dense-1678810861/train	0.7343	0.7361	9	3/14/23, 5:29 PM	7.225 min
3-conv-96-nodes-1-dense-1678810861/validation	0.6564	0.6636	9	3/14/23, 5:29 PM	7.225 min

Figure 10: Legend of Accuracy curves of the best performing models, Train and Validation.

Run	Smoothed	Value	Step	Time	Relative
3-conv-160-nodes-1-dense-1678811835/train	0.02973	0.02962	9	3/14/23, 5:45 PM	7.252 min
3-conv-160-nodes-1-dense-1678811835/validation	1.467	1.441	9	3/14/23, 5:45 PM	7.252 min
3-conv-192-nodes-1-dense-1678812323/train	0.03084	0.0307	9	3/14/23, 5:53 PM	7.28 min
3-conv-192-nodes-1-dense-1678812323/validation	1.222	1.188	9	3/14/23, 5:53 PM	7.28 min
3-conv-224-nodes-2-dense-1678816225/train	0.02818	0.02775	9	3/14/23, 6:58 PM	7.279 min
3-conv-224-nodes-2-dense-1678816225/validation	1.189	1.099	9	3/14/23, 6:58 PM	7.279 min
3-conv-96-nodes-1-dense-1678810861/train	0.03146	0.03128	9	3/14/23, 5:29 PM	7.225 min
3-conv-96-nodes-1-dense-1678810861/validation	1.481	1.49	9	3/14/23, 5:29 PM	7.225 min

Figure 11: Legend of Loss curves of the best performing model, Train and Validation.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 random_flip (RandomFlip)    (None, 180, 180, 3)       0

 random_flip_1 (RandomFlip)  (None, 180, 180, 3)       0

 random_rotation (RandomRota  (None, 180, 180, 3)      0
 tion)

 random_zoom (RandomZoom)    (None, 180, 180, 3)       0

 rescaling (Rescaling)       (None, 180, 180, 3)       0

 conv2d_6 (Conv2D)           (None, 178, 178, 192)     5376

 activation_1 (Activation)   (None, 178, 178, 192)     0

 max_pooling2d_7 (MaxPooling  (None, 89, 89, 192)      0
 2D)

 conv2d_7 (Conv2D)           (None, 87, 87, 192)       331968

 activation_2 (Activation)   (None, 87, 87, 192)       0

 max_pooling2d_8 (MaxPooling  (None, 43, 43, 192)      0
 2D)

 conv2d_8 (Conv2D)           (None, 41, 41, 192)       331968

 activation_3 (Activation)   (None, 41, 41, 192)       0

 max_pooling2d_9 (MaxPooling  (None, 20, 20, 192)      0
 2D)

 flatten_1 (Flatten)         (None, 76800)             0

 dense_2 (Dense)             (None, 192)               14745792

 activation_4 (Activation)   (None, 192)               0

 dense_3 (Dense)             (None, 22)                4246

 activation_5 (Activation)   (None, 22)                0

=================================================================
Total params: 15,419,350
Trainable params: 15,419,350
Non-trainable params: 0
_________________________________________________________________
```

Figure 12: Summary of Model t1.

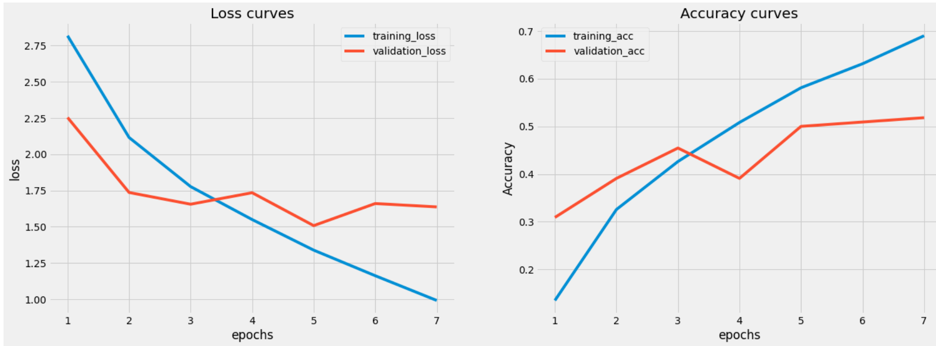

Figure 13: Loss and Accuracy curves of Model t1.

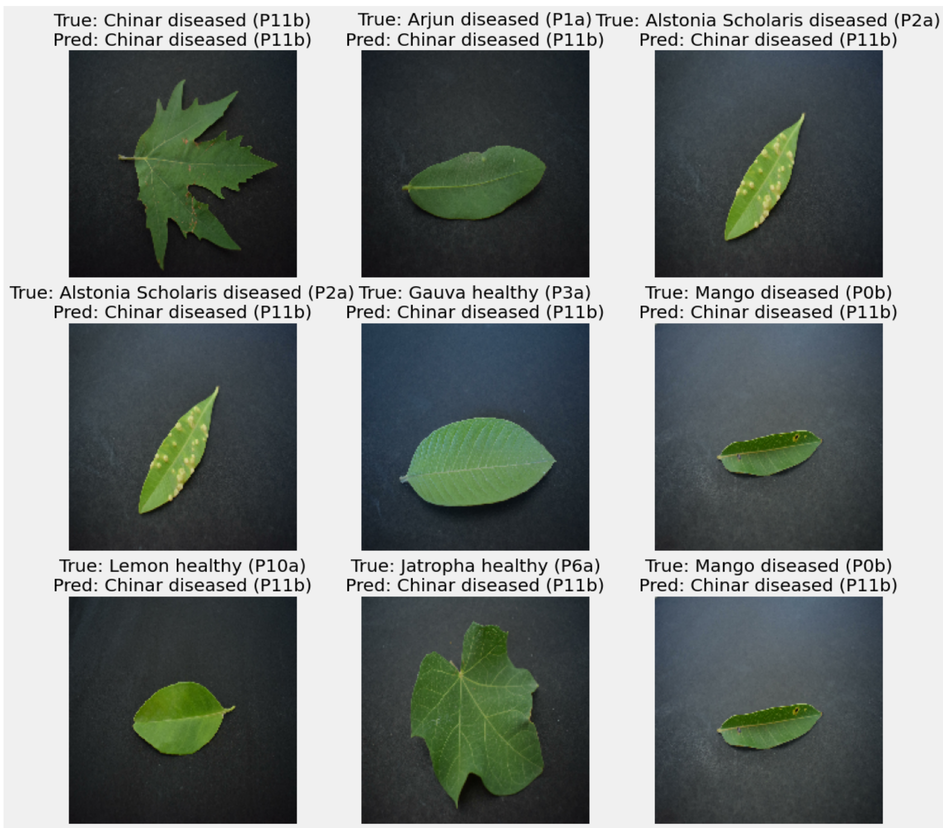

Figure 14: Labels predicted by Model t1 VS the true labels.

3.4 Model q

In *Model q*, the batch size of the train set and the images size was slightly modified. Batch size was set to 50, and image height and width was set to 64. Moreover, also the model architecture changed.

3.4.1 Architecture

Model q is a sequential model. The number of layers is 11 and the parameters in the model are 202.486.

1. **First Convolutional layer**: the size of the filter is 64, the Kernel is 3x3. The input shape is defined (64x64, rgb image), the padding is set to "same", and the activation function is "relu";

2. **Pooling layer**: pooling size is 2x2; item **Batch Normalization layer**: default parameters;

3. **Pooling layer**: pooling size is 2x2;

4. **Second Convolutional layer**: the size of the filter is 160, the Kernel is 3x3, the padding is set to "same", and the activation function is "relu";
5. **Pooling layer**: pooling size is 2x2;
6. **Dropout layer**: dropout rate set to 0.2;
7. **First Dense layer**: the size of the filter is 64, and the activation function is "relu";
8. **Flatten layer**;
9. **Final Dense layer**: the size of the filter is 22;
10. **Activation function layer** with Softmax.

In figure 15 it is possible to see the summary of Model q.

```
Model: "sequential_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_32 (Conv2D)          (None, 62, 62, 96)        2688

 max_pooling2d_37 (MaxPoolin  (None, 31, 31, 96)       0
 g2D)

 batch_normalization_14 (Bat  (None, 31, 31, 96)       384
 chNormalization)

 max_pooling2d_38 (MaxPoolin  (None, 15, 15, 96)       0
 g2D)

 conv2d_33 (Conv2D)          (None, 13, 13, 160)       138400

 max_pooling2d_39 (MaxPoolin  (None, 6, 6, 160)        0
 g2D)

 dropout_8 (Dropout)         (None, 6, 6, 160)         0

 dense_27 (Dense)            (None, 6, 6, 64)          10304

 flatten_12 (Flatten)        (None, 2304)              0

 dense_28 (Dense)            (None, 22)                50710

 activation_10 (Activation)  (None, 22)                0

=================================================================
Total params: 202,486
Trainable params: 202,294
Non-trainable params: 192
```

Figure 15: Summary of*Model q*.

3.4.2 Compiling and fitting

The parameters for the compilation and fitting of the models are the same as for the previous models.

3.4.3 Loss curves and Accuracy curves

The train and validation accuracy curves are much closer to each other as opposed to other models (see Figure 16), and moreover the validation accuracy has a better trend. This is also true for loss curves. It is noticeable that the last epoch is 6. Overall, over-fitting is still observable, and the result could be further improved.

3.4.4 Prediction skills

The f1 metric, for the evaluation of the prediction skills of *Model q*, was 0.7527712186803096, a much better result than the previous. In Figure 17 an example of labels prediciton is displayed. Overall, prediction skills increased from past models.

Figure 16: Loss and accuracy curves of *Model q*.

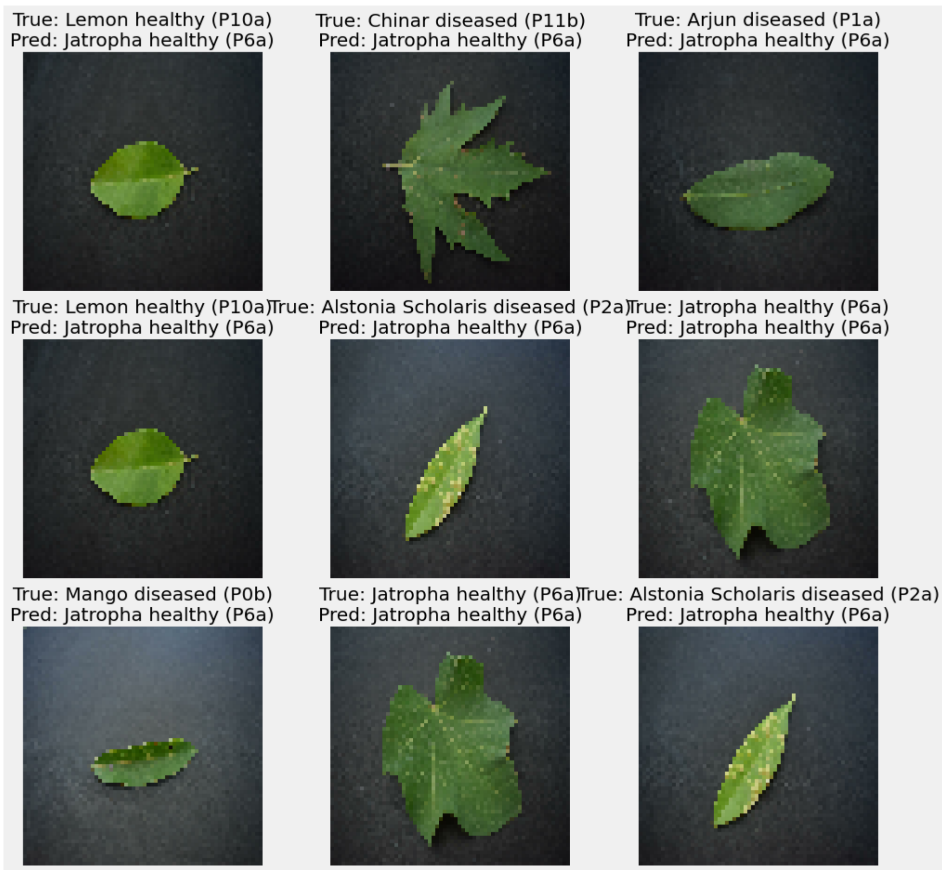

Figure 17: Labels predicted by Model q VS the true labels.

3.5 Conclusions

The best performing model is *Model q*, which has a different data input, and architecture than Model 1, and Model t1.

3.6 Appendix

Extended URL for Tensorboard interactive visualization:https://tensorboard.dev/experiment/E
i23zyBRTnmEbS8pEF7aMw/#scalars&runSelectionState=eyIzLWNvbnYtMTI4LW5vZGVzLTAtZGVuc2U
tMTY3ODgwNzkzOS90cmFpbiI6dHJ1ZSwiMy1jb252LTEyOC1ub2Rlcy0wLWRlbnNlLTE2Nzg4MDc5MzkvdmF
saWRhdGlvbiI6dHJ1ZSwiMy1jb252LTEyOC1ub2Rlcy0xLWRlbnNlLTE2Nzg4MTEzNDgvdHJhaW4iOnRydWU
sIjMtY29udi0xMjgtbm9kZXMtMS1kZW5zZS0xNjc4ODExMzQ4L3ZhbGlkYXRpb24iOnRydWUsIjMtY29udi0
xMjgtbm9kZXMtMi1kZW5zZS0xNjc4ODE0NzYyL3RyYWluIjp0cnVlLCIzLWNvbnYtMTI4LW5vZGVzLTItZGV
uc2UtMTY3ODgxNDc2Mi92YWxpZGF0aW9uIjp0cnVlLCIzLWNvbnYtMTI4LW5vZGVzLTMtZGVuc2UtMTY3ODg
xODE3NC90cmFpbiI6dHJ1ZSwiMy1jb252LTEyOC1ub2Rlcy0zLWRlbnNlLTE2Nzg4MTgxNzQvdmFsaWRhdGl
vbiI6dHJ1ZSwiMy1jb252LTE2MC1ub2Rlcy0wLWRlbnNlLTE2Nzg4MDg0MjQvdHJhaW4iOnRydWUsIjMtY29
udi0xNjAtbm9kZXMtMC1kZW5zZS0xNjc4ODA4NDI0L3ZhbGlkYXRpb24iOnRydWUsIjMtY29udi0xNjAtbm9
kZXMtMS1kZW5zZS0xNjc4ODExODM1L3RyYWluIjp0cnVlLCIzLWNvbnYtMTYwLW5vZGVzLTEtZGVuc2UtMTY
3ODgxMTgzNS92YWxpZGF0aW9uIjp0cnVlLCIzLWNvbnYtMTYwLW5vZGVzLTItZGVuc2UtMTY3ODgxNTI0OC9
0cmFpbiI6dHJ1ZSwiMy1jb252LTE2MC1ub2Rlcy0yLWRlbnNlLTE2Nzg4MTUyNDgvdmFsaWRhdGlvbiI6dHJ
1ZSwiMy1jb252LTE2MC1ub2Rlcy0zLWRlbnNlLTE2Nzg4MTg2NjAvdHJhaW4iOnRydWUsIjMtY29udi0xNjA
tbm9kZXMtMy1kZW5zZS0xNjc4ODE4NjYwL3ZhbGlkYXRpb24iOnRydWUsIjMtY29udi0xOTItbm9kZXMtMC1
kZW5zZS0xNjc4ODA4OTEyL3RyYWluIjp0cnVlLCIzLWNvbnYtMTkyLW5vZGVzLTAtZGVuc2UtMTY3ODgwODk
xMi92YWxpZGF0aW9uIjp0cnVlLCIzLWNvbnYtMTkyLW5vZGVzLTEtZGVuc2UtMTY3ODgxMjMyMy90cmFpbiI
6dHJ1ZSwiMy1jb252LTE5Mi1ub2Rlcy0yLWRlbnNlLTE2Nzg4MTU3MzYvdHJhaW4iOnRydWUsIjMtY29udi0
xOTItbm9kZXMtMS1kZW5zZS0xNjc4ODEyMzIzL3ZhbGlkYXRpb24iOnRydWUsIjMtY29udi0xOTItbm9kZXM
tMi1kZW5zZS0xNjc4ODE1NzM2L3ZhbGlkYXRpb24iOnRydWUsIjMtY29udi0xOTItbm9kZXMtMy1kZW5zZS0
xNjc4ODE5MTQ3L3RyYWluIjp0cnVlLCIzLWNvbnYtMjI0LW5vZGVzLTAtZGVuc2UtMTY3ODgwOTQwMC90cmF
pbiI6dHJ1ZSwiMy1jb252LTE5Mi1ub2Rlcy0zLWRlbnNlLTE2Nzg4MTkxNDcvdmFsaWRhdGlvbiI6dHJ1ZSw
iMy1jb252LTIyNC1ub2Rlcy0xLWRlbnNlLTE2Nzg4MTI4MTMvdHJhaW4iOnRydWUsIjMtY29udi0yMjQtbm9
kZXMtMC1kZW5zZS0xNjc4ODA5NDAwL3ZhbGlkYXRpb24iOnRydWUsIjMtY29udi0yMjQtbm9kZXMtMS1kZW5
zZS0xNjc4ODEyODEzL3ZhbGlkYXRpb24iOnRydWUsIjMtY29udi0yMjQtbm9kZXMtMi1kZW5zZS0xNjc4ODE
2MjI1L3RyYWluIjp0cnVlLCIzLWNvbnYtMjI0LW5vZGVzLTItZGVuc2UtMTY3ODgxNjIyNS92YWxpZGF0aW9
uIjp0cnVlLCIzLWNvbnYtMjI0LW5vZGVzLTMtZGVuc2UtMTY3ODgxOTYzNi92YWxpZGF0aW9uIjp0cnVlLCI
zLWNvbnYtMjI0LW5vZGVzLTMtZGVuc2UtMTY3ODgxOTYzNi90cmFpbiI6dHJ1ZSwiMy1jb252LTMyLW5vZGV
zLTAtZGVuc2UtMTY3ODgwNjQ2Ny90cmFpbiI6dHJ1ZSwiMy1jb252LTMyLW5vZGVzLTAtZGVuc2UtMTY3ODg
wNjQ2Ny92YWxpZGF0aW9uIjp0cnVlLCIzLWNvbnYtMzItbm9kZXMtMS1kZW5zZS0xNjc4ODA5ODg4L3RyYWl
uIjp0cnVlLCIzLWNvbnYtMzItbm9kZXMtMS1kZW5zZS0xNjc4ODA5ODg4L3ZhbGlkYXRpb24iOnRydWUsIjM
tY29udi0zMi1ub2Rlcy0yLWRlbnNlLTE2Nzg4MTMzMDMvdHJhaW4iOnRydWUsIjMtY29udi0zMi1ub2Rlcy0
yLWRlbnNlLTE2Nzg4MTMzMDMvdmFsaWRhdGlvbiI6dHJ1ZSwiMy1jb252LTMyLW5vZGVzLTMtZGVuc2UtMTY
3ODgxNjcxNi90cmFpbiI6dHJ1ZSwiMy1jb252LTMyLW5vZGVzLTMtZGVuc2UtMTY3ODgxNjcxNi92YWxpZGF
0aW9uIjp0cnVlLCIzLWNvbnYtNjQtbm9kZXMtMC1kZW5zZS0xNjc4ODA2OTYxL3RyYWluIjp0cnVlLCIzLWN
vbnYtNjQtbm9kZXMtMC1kZW5zZS0xNjc4ODA2OTYxL3ZhbGlkYXRpb24iOnRydWUsIjMtY29udi02NC1ub2R
lcy0xLWRlbnNlLTE2Nzg4MTAzNzQvdHJhaW4iOnRydWUsIjMtY29udi02NC1ub2Rlcy0xLWRlbnNlLTE2Nzg
4MTAzNzQvdmFsaWRhdGlvbiI6dHJ1ZSwiMy1jb252LTY0LW5vZGVzLTMtZGVuc2UtMTY3ODgxNzIwMi92YWx
pZGF0aW9uIjp0cnVlLCIzLWNvbnYtNjQtbm9kZXMtMy1kZW5zZS0xNjc4ODE3MjAyL3RyYWluIjp0cnVlLCI
zLWNvbnYtNjQtbm9kZXMtMi1kZW5zZS0xNjc4ODEzNzg4L3RyYWluIjp0cnVlLCIzLWNvbnYtNjQtbm9kZXM
tMi1kZW5zZS0xNjc4ODEzNzg4L3ZhbGlkYXRpb24iOnRydWUsIjMtY29udi05Ni1ub2Rlcy0wLWRlbnNlLTE
2Nzg4MDc0NDkvdHJhaW4iOnRydWUsIjMtY29udi05Ni1ub2Rlcy0wLWRlbnNlLTE2Nzg4MDc0NDkvdmFsaWR
hdGlvbiI6dHJ1ZSwiMy1jb252LTk2LW5vZGVzLTEtZGVuc2UtMTY3ODgxMDg2MS92YWxpZGF0aW9uIjp0cnV
lLCIzLWNvbnYtOTYtbm9kZXMtMS1kZW5zZS0xNjc4ODEwODYxL3RyYWluIjp0cnVlLCIzLWNvbnYtOTYtbm9
kZXMtMi1kZW5zZS0xNjc4ODE0Mjc0L3RyYWluIjp0cnVlLCIzLWNvbnYtOTYtbm9kZXMtMi1kZW5zZS0xNjc
4ODE0Mjc0L3ZhbGlkYXRpb24iOnRydWUsIjMtY29udi05Ni1ub2Rlcy0zLWRlbnNlLTE2Nzg4MTc2ODcvdHJ
haW4iOnRydWUsIjMtY29udi05Ni1ub2Rlcy0zLWRlbnNlLTE2Nzg4MTc2ODcvdmFsaWRhdGlvbiI6dHJ1ZX0
%3D.

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.